INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(54) Title: DATABASE SYSTEMS HAVING SINGLE-ASSOCIATION STRUCTURES AND METHOD FOR SEARCHING DATA IN THE DATABASE SYSTEMS

(57) Abstract

A database system is comprised of single-association databases (704, 705, 706) having combined two data lines, database drivers (712, 713, 714) to execute the searching of single-association databases and interface driver (720) to combine the search results of database drivers (712, 713, 714). A user can obtain desired search result by using interface driver (720) which selects a single-association database having desired search field, executes the search, and combines each of generated search results.

DESCRIPTION

DATABASE SYSTEMS HAVING SINGLE-ASSOCIATION STRUCTURES AND METHOD FOR SEARCHING DATA IN THE DATABASE SYSTEMS

5

FIELD OF THE INVENTION

The present invention relates to a database system, and more particularly relates to database system in which data is organized in a plurality of single-association structure.

10

BACKGROUND OF THE INVENTION

Databases are among the most important assets of a company. Examples of databases are customer records (e.g., name, address, phone number, credit card number, etc.), accounting data (e.g., account receivable

15   and account payable), and mailing lists (for soliciting new customers).   A company would plunge into financial crisis if its databases are destroyed.   For example, the company will not be able to collect its bills if its account receivable files are erased.   Thus, the value of the databases of a company could be higher than the value of its tangible assets.

20   A database is a collection of data organized in a predetermined manner. Typically, the data is arranged into a plurality of records having identical structure.   Each record contains one or more fields (e.g., name, address, and phone numbers) for holding appropriate data.   In order to retrieve data quickly, a database is typically associated with an index which contains a table of keys

25   representing every record in the database.   The keys typically correspond to the data in a predetermined field (e.g., name) of the records.   The keys are arranged in a user-defined criterion (e.g., ascending or descending alphabetical order).   Generally, records are stored in the database in the order they are entered.   In most cases, records are entered without any particular

30   order, and can be considered random.   However, the index is re-arranged in accordance with the criterion every time when one or a set of records are entered.   When it is desirable to locate a record, the index is searched first. Because the index is well-organized, the search time is very short.   The correct record can be easily found based on the information obtained from the

35   search result.   If the index in not used, the entire database (which could be

many times the size of the index) needs to be searched.

The sizes of these databases are typically very big (several million bytes). The number of rows and columns are very large. There could be many indexes associated with a database. Further, the search "engines" are very
5 complex because they have to handle the huge databases and work with all the indexes. As a result, even in the age of personal computers, these databases are best handled by mainframe computers.

There are "free-form" databases in which the data is not organized in any predetermined manner while allowing every words in the database to be
10 searched. This type of databases are often used by on-line information providers which provide full-text searching capability to newspapers, magazines, court decisions, etc. In this type of databases, all the words in the text are indexed. The "free-form" databases allow for efficient retrieval of data, but is typically not flexible and very inefficient in utilizing computer resources
15 (e.g., the index files are typically very large).

Thus, there is a need to design a simple, fast, flexible and efficient database system.

SUMMARY OF THE INVENTION

20 The present invention involves a novel database system comprising a plurality of single-association databases each associated with a database driver, and a method for searching data in the database system. A single-association database contains a plurality of records, each record associates one piece of data with another piece of data. One implementation of the
25 single-association database is an ASCII file containing many lines, each line associate one set of ASCII characters with another set of ASCII characters. An example of such a file is:

A111 = John

A113 = Peter

30 B111 = Mary

In this example, the symbol "=" is an association symbol, and the set of ASCII characters to the left of the "=" sign (e.g., "A111") is associated with the set of characters to the right of the "=" sign (e.g., "John").

The database driver is a software routine optimized for searching
35 information in its associated database. As a result, the size of the driver is

small and the search speed is fast. Further, the plurality of single-association databases can be combined in a variety of ways so that complicated searches can be easily conducted. These combinations give rise to a very flexible database system.

5   The present database system does not need to have index files. The structure of each database in the present system is often simpler than an index file of a conventional database system. As a result, there is no need to construct an index file to speed up searching. The lack of index files is another advantage of the present database system.

10   In this database system, all the database drivers and their associated databases are hierarchically equal. Architecturally, these drivers and databases are independent of each other. Adding more databases will not increase the complexity of the system, although the information contained in the database system increases. As a result, the size of the present database
15   system can be increased without penalizing its performance.

The database system of the present invention can be easily implemented on a network (both local and wide area). The databases and the drivers could reside on different computers of a network. A computer can combine its existing databases and drivers with the databases and drivers downloaded
20   from other computers so as to build a desired database system comprising a plurality of single-association databases.

The database system of the present invention can be implemented using a novel bossless computer program architecture (called "digital cell technology") comprising a plurality of program modules called "cells." Under this
25   architecture, each cell is hierarchically equal, i.e., there is no controlling (or boss) cell. An application can start from any cell, and can terminate at any cell. Typically, many cells are executing either sequentially or concurrently. Various applications can be designed by controlling the operation of these cells. As explained above, the database system of the present invention comprises a
30   plurality of database drivers which are hierarchically equal. This structure is compatible with the digital cell technology. In this embodiment, the database and interface drivers could be implemented as cells.

Each cell is associated with a file, called a DNA file. The characteristics and operation of the cells are determined by their associated DNA files. Cells
35   communicate by writing statements to the DNA files associated with other cells using a protocol called digital shifting function (DSF). Once written, the origin

of these DSF statements is ignored.   There is no need to "return" to the cells which originate the statements.   Further, the DSF statements are executed by the cells without regard to their origins.

5   The cells execute statements in their associated DNA files.   These statements are executed sequentially.   The cells retain full control of the execution, i.e., there is no need to turn over execution to other cells during or after the execution of statements.   There is no need to report to other cells on the status or results of execution.

10   The digital cell technology does not require the preservation of linkage information because there is no need for the cells to return information or control to other cells.   This technology is flexible because each cell is hierarchically at the same level as other cells, thus can be selected to perform a certain task based only on its ability instead of on its hierarchic level.   For example, when there is no need for a cell to exist, it can be removed from an

15   application.   On the other hand, program modules in conventional boss architecture cannot be removed arbitrarily.   For example, the main program cannot be removed, even if it is not performing any useful function.

In one embodiment of the database system, the single-association databases could be embedded in the DNA files of the associated database

20   drivers.   In this way, the database drivers are integrated with their associated databases.   Some of the statements executable by the database drivers relate to searching of their associated databases.   Thus, other cells can request a database driver to search its associated database by sending a search statement to the DNA file of the database driver.   The requesting cells can

25   also specify the destination of the search result.   The destination could be the requesting cell's own DNA file, the driver cell's DNA file, or an independent file. Thus, the present database system can be easily designed to interact with non-database related cells.   This level of flexibility allows more efficient and powerful database systems to be designed.

30   In applications formed by a plurality of cells (including database drivers, requesting cells and other cells), each cell is at the same level as the other cells.   Thus, each database driver and requesting cell can execute its designed functions independent of each other and can interact with any other cells it desires.   Each cell can issue statements to another cells without going

35   through a chain of cells.   The mode of interaction of the present invention is direct (i.e., from one cell to another directly instead of through a chain of modules).   As a result, the structure of an application is simple and the

execution speed is fast.

These and other features and advantages can be understood from the following detailed description of the invention together with the accompanying drawings.

5

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a schematic diagram of a database system of the present invention.

Fig. 2 is an exemplary graphic display of a database driver of the present
10   invention.

Fig. 3 is an exemplary graphic display of an interface driver of the present invention.

Fig. 4 is a schematic diagram of the database system of the present invention implemented on a network system.

15   Fig. 5 is a schematic diagram of an application based on a boss less architecture which can be used to implement the drivers of the present invention.

Fig. 6 is a drawing showing the structure of an application using a digital cell technology.

20   Fig. 7 is a block diagram showing the logical structure of a DNA file associated with a cell CA.

Fig. 8 is a block diagram of the logical structure of the cell CA.

Fig. 9 is a block diagram showing the logical structure of a DNA file associated with a database driver cell CB.

25   Fig. 10 is a drawing showing a flow of DSF statements when results of searching databases are combined.

Fig. 11 is a block diagram of a computer system which can be used to run the database system of the present invention.

30   ## DETAILED DESCRIPTION OF THE INVENTION

The present invention is directed to a novel database system.   The following description is presented to enable any person skilled in the art to make and use the invention.   Descriptions of specific applications are

provided only as examples. Various modifications to the preferred embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus,

5   the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

Fig. 1 is a schematic drawing of a database system 700 of the present invention. Database system 700 contains a plurality of single-association

10   databases, such as databases 704-706. One way to picture a single-association database is a table having many rows and two columns. The two columns in each row associate one set of data with another set of data. The data could be text, numbers, graphic, audio and video data. In an example involving a library database system used to keep track of all books in a library,

15   the first column of database 704 could be the identification (ID) number of all books in the library while the second column could be the year of publication of each book. Thus, database 704 shows a single association between a pair of data. This is different from conventional database wherein each record contains many fields, thereby associating the data in one field with the data in

20   many fields.

In order to be useful, many single-association databases are needed in system 700. Thus, another database (e.g., database 705) may associate the ID with the authors of the books while a third database (e.g., database 706) may associate the ID to the names of the persons who are borrowing the books.

25   It is possible for the first column of the databases in system 700 to be different. For example, the first column of one of the databases in the above example may be the title of the books instead of the ID number. However, if it is desirable to link this database to the other databases, there is a need to have a database which link the ID to the title of the books.

30   Each database is associated with a database driver. The main function of the database driver is to perform searches on its associated database and return the results of the searches to an appropriate file. The database driver could also perform other functions on its associated database, such as editing the records in the database. In Fig. 1, databases 704-706 are associated with

35   drivers 712-714, respectively.

In a conventional database system, a single driver (i.e., search routine) is associated with all the complex databases in the system. Thus, if a database

system contains ten separate databases, the same driver would be used to perform search on the ten databases. Because the driver needs to handle all the requirements of a large number of complex databases, the size of the driver becomes very large. This is different from the database system of the present
5 invention in which each driver is specially designed to operate on its associated database.

Fig. 2 shows an exemplary graphic display 740 optionally associated with database driver 712. The underlying database 704 provides the relationship between the ID number and the year of publication of books in the above-
10 mentioned library. Display 740 contains a window 742 allowing a user to enter a relationship. In Fig. 2, the relationships are greater than (>), less than (<), equal to (=), greater than or equal to (>=), and less than or equal to (<=). It should be appreciated that other relationship can be included. Display 740 also contains a window 744 showing the set of possible data in the column (in
15 this case, all the years within a range). Window 744 contains a scroll bar 745 allowing the user to scroll through all the available years. The user can define a search criterion by clicking on a relationship (e.g., greater than) in window 742 and a year (e.g., 1960) in window 744. If it is desirable to define a range, display 740 provides an "AND" button 746 for defining an AND relationship.
20 Thus, the user can click on AND button 746, and then select another relationship (e.g., less than) in window 742 and a year (e.g., 1980) in window 744. The user can then click on a SEARCH button 748 to execute the search. The result of the search is display on a window 750. Window 750 contains a scroll bar 751 allowing the user to review all the books which meet the selection
25 criterion.

To make the display more user friendly, display 700 may contain a window 752 showing the search criterion. Display 740 may contain other buttons, such as a CANCEL button 754 or a button for forming an "OR" operation (or other logic variations) of the search relationships. For example, if the user
30 wishes to search for books which are published in 1960 and 1980, the user can define the search relationship (using the above described windows) and then click on the "OR" button.

Typically, an application needs to perform a search on more than one databases. For example, the application may like to search for all books
35 published by a certain publisher between 1960 and 1970. This search would involve (i) a search on a database associating the ID with the publishers and (ii) a search on a database associating the ID with the years of publication. The

results of the two searches are then "ANDed" together.

Returning now to Fig. 1, an interface driver 720 is used to manage searches on multiple databases. One of the functions of an interface routine 720 is to allow an application 724 to specify the databases to be searched and
5   the search criteria.

Fig. 3 shows an exemplary display 770 optionally associated with interface driver 720. Display 770 contains a plurality of windows (such as windows 772-775), one for displaying information relating to a selected database. Each one of windows 772-775 has a similar structure as display 740 of Fig. 2,
10   and consequently will not be shown in detail. The user can select search criteria in each window using the methods described above. The result of the search for each database is displayed in the corresponding window. The user can then click on one of the logic relationships shown in a window 778 to select a search for the results of all the databases. A click on a SEARCH button 782
15   starts the search. The final result is displayed on a result window 780. For example, if the "AND" row in window 778 is selected, window 780 would show the AND of the search results in windows 772-775.

Display 770 may contain other buttons, such as a CANCEL button 784.

The present invention is different from prior art database systems which
20   contain a small number of multi-association databases and a search routine. Typically, the structures of the prior art databases are very complex because each database is designed to contain as much information as possible. As a result, the search routine is also very complex because it has to understand the complicated structure of the underlying database. Consequently, the
25   databases and search routine are very difficult to use and maintain.

The database system of the present invention is especially useful in a distributed environment over a local or wide area network. These networks comprise a plurality of computers. Some of the computers can be used to develop databases and associated drivers of the present invention. Other
30   computers can download and use these databases. An example of a wide-area network is the Internet.

Fig. 4 is a schematic diagram showing a network system 800 of the present invention. System 800 comprises a plurality of computers 802, 804, 806, 808 and 810. These computers are coupled to a network 812, which could be a
35   local or wide area network. A database (and its associated driver) 820 has been developed by and is stored in computer 802. Two databases 824 and

825 reside in computer 804. Computer 804 can download database 820 from computer 802 using a conventional network protocol. The downloaded database 820A is shown in Fig. 4 as dashed lines. Applications in computer 804 can now use all three databases, including 820A downloaded from

5   computer 802. Similarly, computer 808 can download database 820 from computer 802 using a conventional network protocol. The downloaded database 820B is shown in Fig. 4 as dashed lines. Applications in computer 808 can now use an existing database 828 and database 820B downloaded from computer 802.

10   It should be noted that a database and its associated driver could reside on different computers on the network. For example, a database could reside on computer 806 while its associated driver could reside on computer 810. A computer wishing to use the database needs to download the database and the associated driver from computers 806 and 810.

15   Because the structure of the databases and drivers of the present invention could be very simple, even low performance personal computers and relatively unskilled users could design valuable databases. These databases and drivers can be easily downloaded and used by other computers in the network. Further, only those databases and drivers that are actually used by

20 a computer need to be downloaded, thereby reducing traffic on the network. On the other hand, designing conventional databases is a complex task that can be handled only by experienced database designers using powerful computers. Transferring a conventional database on a network requires considerable bandwidth because the size of a conventional database is

25 typically very large.

The database system of the present invention contains single-association databases and related drivers. The databases are very flexible, and can be changed and added easily. The drivers are designed to work with their associated database. Because the underlying databases have a simple

30 structure, the drivers also have a simple structure and cam be designed very efficiently.

One aspect of the database system of the present invention is to reduce the underlying structure to a simple form. This aspect is compatible with a new computer software architecture called "digital cell technology." As a

35 result, the database of the present invention can best be implemented using the digital cell technology. A detailed description of this technology has been disclosed in copending the United States patent application (Serial No.

08/539,806) filed October 5, 1995 and corresponding International patent application (Serial No. JP 96/00821) filed March 28, 1996. These copending patent applications are hereby incorporated by reference.

5   An implementation of the present inventive database system using digital cell technology is now described. One characteristic of the digital cell technology is that it is a "bossless" architecture because every program module (or cell) is on equal footing with other program modules. There is no module that controls the overall operation of the program (i.e., no boss).

10   Fig. 5 is a schematic view of an application 160 based on the "bossless" digital cell technology. Application 160 comprises a plurality of program modules, such as modules 162-165. Each program module (called a "cell") is the same as the other cells in an hierarchical sense. Cells are linked together in a way in which no history or linkage information needs to be retained. Each link is independent. For example, there is no need for links to be active 15 simultaneously. Each link is direct, i.e., two cells can be linked directly without the need of using one or more intermediate links. For example, cells 162 and 164 can be linked directly using line 166 instead of using lines 167 and 168 and passing through an intermediate cell. An application can be formed by defining the cells involved and using the direct link.

20   Fig. 6 is a drawing showing the structure of an application 200 using the bossless architecture of the present invention. Application 200 contains a plurality of cells, labeled as C1-C5, loaded and executing in RAM. Each cell has an associated file (labeled as D1-D5), called DNA file, which contains information of the cell. The term "DNA" is used here in analogy with the 25 biological relationship between a living cell and its DNA. At a desired time, cell C1 can send statements (called "DSF statements") to cell C2 using a protocol called digital shifting function ("DSF") protocol. Cell C2 will execute these statements. The detail structures of cells, DNA files and the DSF protocol will be described below.

30   In this application 200, Cell C2 does not retain information on the origin of these statements, i.e., no history of the inter-process communication is kept. Thus, once cell C1 completes its writing of DSF statements to cell C2, there is no further linkage between cells C1 and C2. Cell C2 does not know the origin of these statements during their execution. It is possible for cell C1 to later 35 establish communication with cell C2 again by sending another set of statements to C2. However, this communication is separate from the previous communication, and terminates once the new set of DSF statements is sent.

Each of the cells can send DSF statements to any of the cells it desires. Thus, cell C1 can also send statements to cell C3.   Similarly, cell C2 can send statements to cell C4, which in turn can send statements to cell C1.   Cell C3 can also send statements to cell C1.

5   In this example, cells C1 and C2 are not bosses to C4.   For example, when C4 is executing DSF statements, there is no need to maintain any links between cells C1 and C2 and between cells C2 and C4.   Cell C4 has no obligation to report results of execution to any cells in application 200.   Links are maintained only during the time DSF statements are transferred.   Further, 10   the writing of statements by cell C1 to cell C2 could be independent of the writing of statements by cell C2 to cell C4.   In addition, cell C4 merely execute statements, and does not care where the statements come from.

In this architecture, there is no need to save and restore register values on a stack when cell C2 executes statements written by cell C1.   There is no need 15   to register cells in a central database prior to sending commands.   There is no need to send messages back and forth to report status of execution.   As a result, the application can be executed quickly.

As explained below, a cell can be implemented as an "EXE" file (in the MSDOS or MS Windows environment) and can be invoked by loading into RAM 20   for execution using well known methods in accordance with the operating environment.   The cell's associated DNA file can also be loaded into RAM. The invoked cell takes on the attributes stored in its DNA file. It is also possible to modify the DNA file when the cell is invoked or while running by writing to the file (which could be an ASCII file).

25   Fig. 7 is a block diagram showing the logic structure of a DNA file 250 associated with a cell, such as cell CA.   File 250 has a section 252 containing parameters ("own parameters") related to the characteristics of cell CA itself. For example, section 252 may contain parameters related to the way cell CA manifest itself when invoked: the window size and background color of cell CA, 30   the name of cell CA, the names of audio files associated with its invocation and termination, etc.

File 250 also contains a section 254 containing linking parameters ("link parameters") on cells related to cell CA.   Examples of the parameters contained in this section are: the names, symbols and positions of the other 35   cells.   One of the parameter is "close," which is interpreted as closing cell CA when the cell associated with this parameter is invoked.

File 250 further contains a DSF information section 256.   This section contains a regular section 257 and a top priority function section 264.   The structure of the regular section 257 and top priority function section 264 are substantially the same, except that top priority function section 264 has a
5   higher priority in DSF statement execution.   These two sections contain individual headers for identifying the sections (e.g., each section headed by a different name or symbol).

Regular section 257 contains a "condition" section 258 and a statements section 260.   Statements section 260 comprises statements sent to cell CA by
10   other cells.   Statements in statements section 260 are executed sequentially. Each   statement also contains parameters necessary for the execution of the statement.   Condition section 258 comprises three components: (a) a first pointer to the last DSF statement currently existing in statements section 260, (ii) a second pointer to the current DSF statement being processed by cell CA,
15   and (iii) the current status of the cell.

Top priority function section 264 contains a condition section 266 and a command lines section 268.   The structure of condition section 266 is similar to the structure of condition section 258.   Command lines section 268 contains executable command lines which are sent by other cells using DSF (or similar)
20   protocol.   The command lines have a higher execution priority than the statements in statements section 260.   The command lines in command lines section 268 are executed sequentially.

It should be appreciated that the logic structure shown in Fig. 7 can be implemented using one or more physical files.   Further, portions of the logical
25   sections may intermingle physically.   In one embodiment, DNA file is a text file. Thus, the content of the DNA file can be modified by using a regular text editor.

Statements sent by one cell to another follow the DSF protocol.   A sending cell (e.g., cell CS) sets up a communication link with the DNA file 250 associated with cell CA.   Specifically, it looks up the address of DNA file 250
30   and determines whether DNA file 250 is able to accept DSF statements by examining its status in condition section 258.   Statements will be issued by cell CS only when cell CA is ready to accept them.   In one embodiment, the issuance of statements involves writing ASCII characters to statements section 260 of DNA file 250.

35   When cell CS is authorized to issue statements to cell CA, cell CS reads the first pointer (in condition section 258) to the last DSF statement to

determine the appropriate address to write the DSF statements.   It is important not to overwrite DSF statements already existed in cell CA.   Cell CS writes DSF statements to statements section 260 of DNA file 250.   Cell CS also updates the first pointer in condition section 258 so as to point to the last DSF
5   statement newly written into statements section 260.   The communication link between cells CA and CS is terminated.   It can be seen that cell CA and DNA file 250 do not maintain record (i.e., history) indicated that these new statements originate from cell CS.

It should be appreciated that the above described DSF protocol is only an
10   exemplary protocol.   Other protocol could be used to write DSF statements to cells.   For example, different pointer structures can be used, e.g., the first pointer can point to the position after the last DSF statement.   Different types of status and different ways for checking status information can be used. Further, the statements could be stored in accordance with a logic structure
15   instead of stored physically in a sequential manner. , For example, the statements could be organized into groups with the address of each group pointed to by a pointer.

Command lines are sent by one cell to another using a protocol substantially the same as the DSF protocol.   Because regular statements
20   section 257 and top priority function section 264 have different headers, the sending cell can distinguish between these two sections and write to the appropriate section. Other means for identifying these two section can also be used.

Fig. 8 shows the structure of cell CA.   It is grouped logically into a plurality
25   of sections, each implemented using instructions executable by a computer. Cell CA contains an initialization section 312 and a DNA interface section 314. DNA interface section 314 allows cell CA to read from and write to its corresponding DNA file 250.   Initialization section 312 takes care of housekeeping tasks when invoked, including reading parameters from "own
30   parameters" section 252 of DNA file 250.   Cell CA also contains a DSF interface section 316 (for processing DSF protocol).   This section contains code (or program instructions) for sending and receiving statements/command lines using the DSF protocol.

Cell CA contains an execution section 318 containing code for
35   automatically executing statements and command lines in DNA   file 250 written by other cells.   The code sequentially read and execute statements in statements section 260 of DNA file 250.   After each statement is executed, cell

CA branches to top priority function section 259 and executes all the command lines therein. Cell CA then executes the next statement in statement section 260.

Cell CA contains a temporary memory section 322 for storing temporary information. As an example, it is possible to change attributes (e.g., background color and the size of the display window) of cell CA during its execution. In one embodiment, the changed attributes are temporarily stored in temporary memory section 322 instead of immediately being written to DNA file 250. In this embodiment of cell CA, the attribute information stored in temporary memory section 322 is written into "own parameters" section 252 of DNA file 250 only when cell CA is terminated.

Cell CA also contains a cell invocation section 324 for invoking other cells. In one embodiment, this section obtains information about the cell desired to be invoked and pass this information to a specialized cell which actually invoke the desired cell. It is also possible to incorporate the functionality of this specialized cell in the cell invocation section 324 of cell CA and other cells.

It should be appreciated that the above mentioned sections in cell CA are grouped logically, and portions of these sections could intermingle physically.

It can be seen from the above described structures of cell CA and its associated DNA file 250 that both cell CA and DNA file 250 do not keep track of the origin of the DSF statements. A cell may accept DSF statements (stored in its associated DNA file) from many cells. After the DSF statements have been received, the linkage between the originating and destination cells is terminated. The cell executes DSF statements contained in its associated DNA file without knowledge of how the statements arrive the DNA file. As a result, there is no need to "return" to any cell. As pointed out above in connection with the destination file, nothing in this architecture prevents the recipient cell from returning information to the originating cell.

Typically, the size of each cell is small and the function of the cell well defined. Consequently, the execution speed is fast. As a result of the small size and specialized function, the cells can be easily written to fully utilize the resources of a computer. The communication between cells using DSF is direct, with minimum amount of access to the operating system on which an application is run. As a result, the efficiency is high.

The architecture of the digital cell technology comprises at least two cells which can communicate with each other. The cells are encapsulated program

modules that are specialized for their designed tasks. Therefore, applications developed using this architecture comprise of multiple executable which can run independently or concurrently. The cells interact with each other using the inventive DSF protocol. Each cell can control the action of other cells.

5 For example, a first cell can control a second cell, and the second cell can control the first cell. Therefore, no single cell has complete control over the other cells, or in other words, there is no boss. Another unique characteristic of this architecture is that the cell that receives a command does not keep any information of where the command came from. This lack of historical

10 knowledge allows cells to move forward instead of going backward in a link.

The above described feature of the digital cell technology is especially advantageous in implementing the database system of the present invention. As explained above, the present database system contains a plurality of database drivers designed to serve their associated database. These drivers

15 are implemented as cells. Because there is no overhead in linking, the database system can support a large number of drivers without affecting the performance of the system. As a result, the digital cell technology provides an efficient platform to support the database the database system of the present invention. These two inventive technologies (digital cell and single-

20 association database technologies) form a synergical.cal combination that can outperform conventional database.

In the database system of the present invention, database drivers 712-714 and interface driver 720 can be implemented as cells. These cells can interact with other cell in the system in the manner described above. For

25 example, database drivers 712-714 can receive "search" DSF statements from other cells for initiating a search on their associated database.

The digital cell technology is especially suitable for a networked database environment described above in Fig 4. The driver cells and the associated databases can be obtained from any source, including a remote computer in a

30 network. Once downloaded to a local computer, they can interact with other cells in the local computer. The downloaded cells behave just like local cells.

It should be pointed out that the same database driver cell can run at different times, and access the same or different database at each time. For example, cell A can access database number one at time one and access

35 database number two at time two, and then return to database number one at time three.

In one embodiment of the invention, the database of a database driver cell is stored in its associated DNA file.   If cell CB is a database driver, file 250 contains a section 262 containing the associated database data.

5   Fig. 9 is a diagram showing the logical structure of DNA file 250 associated with driver cell CB.   DNA file 250 comprises database section 262 having a database associated with driver cell CB.   If the size of the database is large, the size of the database section also become large.   For example, the contents of database section 262 is represented as shown below:

                AC102 = John Smith

10   AC103 = Steve Dole

                AC105 = Mike King

In this example, the symbol "=" is used for specifying association.   The data to the left of the "=" sign (i.e., AC102, AC103 and AC105) represent ID numbers of books.   On the other hand, the data to the right of the "=" sign represent

15   author's name.   These data could be the set of ASCII characters.   The number of lines is determined in accordance with the amount of information in the database.

        It is possible that database associated with cell CB be stored in other locations.   In this case, file 250 contains a line which indicates the location of

20   the associated database file.

        If each of single-association databases 704 to 706 for library database system described with reference to Fig. 1 is accomplished as DNA file 250, data (ID and published year, ID and author's name, and ID and borrower's name) of each of single-association databases 704 to 706 are described on

25   database section 262 of each DNA file.

        Top priority function section 264 and regular section 257 of driver cell CB mainly receive DSF statements which perform instructions such as searching database and deleting the record in the database.

        These DSF statements may comprise various parameters such as a

30   search criteria parameter and a destination file parameter (for indicating the location of the file which used to store the results of the search).   DSF information section 256 may include search key section 271 (for accepting search criteria) and destination file section 273.   Other cells can write to these two sections as a part of a search request.   The destination file could be the

35   DNA file of the cell which requests the search, or the DNA file of the current

database driver, or another file.

For example, two statement, "search for ID linked to author John Smith: destination is cell CD" and " search for ID linked to author Mike King: destination is cell CD," exist in statement section 260, and command line

5   "search for ID linked to author Steve Dole: destination is cell CE" is written on command line section 268 after starting of execution of statement "search for ID linked to author John Smith: destination is cell CD."

Cell CB reads statement "author searches for ID of John Smith: destination is cell CD" of statement section 260, then searches the database in database

10   section 262 for the record including "John Smith." When cell CB detects associated record, cell CB reads ID "AC102" of the record and sends "AC102" to cell CD (not shown) indicated by destination file parameter.

Cell CB looks command line section 268. Because command line "search for ID linked to author Steve Dole: destination is cell CE" exists in

15   command line section 268, cell CB searches the database for the record including "Steve Dole." When cell CB detects associated record, cell CB reads ID "AC103" and sends it to cell CE (not shown) indicated by destination file parameter.

Then, cell CB goes back to statement section 260 and executes next

20   statement "search for ID linked to author Mike King: destination is cell CD." When cell CB detects the record including "Mike King" in the database, cell CB reads ID "AC105" and sends it to cell CD indicated by destination file parameter.

In this embodiment, if command line in command line section 268 is

25   "delete the record linked to author Mike King," cell CB deletes the record "AC105 = Mike King" in database section 262. In this case, If statement "search for ID linked to author Mike King: destination is cell CB" is executed, the message saying "There is no associated data" is sent to cell CB.

The search results of a plurality of databases can be combined by using an

30   interface cell. The flow of DSF statement in this case now will be described with reference to Fig. 10.

For example, driver cell E1 comprising data including ID and author's name of a book and driver cell E2 comprising data including ID and published year of a book exist. And interface cell G1 which combines the search results

35   of these driver cells E1 and E2 exists.

When you search for ID of a book whose author's name is "Mike King" or

for ID of a book whose published year is "1970", interface cell G1 sends DSF statement "search for ID linked to author Mike King: destination is cell G1" to DNA file 250 in driver cell E1 (L1), and sends DSF statement "search for ID linked to the published year 1970" to DNA file 250 in driver cell E2 (L2).

5   DSF statement sent from interface cell G1 are written on each DNA file 250 of cell E1 and cell E2, and executed.   Cell E1 and cell E2 send the search results to DNA file 250 of interface cell G1 (L3, L4).   When the search results are written on DNA file 250 of interface cell G1, interface cell G1 executes the OR processing to these search results and obtains the final search results (that

10   is, ID of a book whose author's name is "Mike King" and ID of a book whose published year is "1970").

In case of searching for ID of a book whose author's name is "Mike King," and besides, whose published year is "1970," interface cell G1 executes the AND processing to the search results received from each of driver cells E1 and

15   E2.

It can be seen from above that the digital cell technology is especially suitable for implementing the database system of the present invention.   This database system comprises many database drivers, only few of them may be used at a time.   Under the digital cell technology, all the drivers are

20   implemented independently (instead of subroutines, as is the case in conventional architecture).   Only those drivers that are actually being used are active.   As a result, there is little, if any, reduction in performance by adding more drivers (and their associated database) to the system.

One of the embodiments of the present invention is an application

25   development system which runs under Microsoft's MS Windows.   In this environment, cells are programs stored as ".EXE" files and typically shows a window on a computer monitor when invoked.   By linking these cells, a users can construct an application software just like assemble blocks.   Each cell, with its specific function, is given another function or value through DSF

30   protocol with other cells to produce a variety of applications.

Fig. 11 shows a block diagram of a computer system 600 which can be used to run the database system of the present invention.   Computer system 600 comprises a computer 602 having a central processing unit (CPU) 604 and system memory 606, which could be random access memory (RAM) or read-

35   only memory (ROM), coupled to a system bus 608.   Computer 602 also contains a peripheral bus controller 612 for controlling a peripheral bus 614.

Depending on the architecture of computer 602, peripheral bus 614 could be a
PCI bus, VESI local bus, ISA bus, or other similar buses.   Peripheral bus 614
allows peripheral boards to be connected to computer 602.   Examples of
peripheral boards are a video board 616, a serial board 620 and a data transfer

5   board 622.   CPU 604 and RAM 606 can communicate with the peripheral
boards through peripheral bus controller 612.

Serial board 620 allows serial communication between computer 602 and
one or more external serial devices, such as a mouse 636.

Video board 616 contains circuits to control a monitor 630 and display
10   images thereon.   Video board 616 also contains memory (not shown)
associated with such display.   The memory is preferably a special kind of
memory integrated circuit device, called video RAM (VRAM), designed for
video applications.   The circuits draws images on monitor 630 based on
information stored in the memory.   The images are on monitor 630 are
15   updated at predetermined time intervals.   An example of an image is display
740 shown in Fig. 2A.

If computer system 600 is used to run programs in a windows-based
environment, one or more windows, such as a windows 632 and 634, could be
displayed on monitor 630.

20   Disk controller board 622 is connected to a hard disk 638 and a floppy disk
driver 638.   The MS Windows and the operating system are generally stored
in hard disk 638.   The cells can be stored in floppy diskettes and downloaded
into hard disk 638.   In one embodiment of the present invention, individual
database driver and its associated database can be designed and stored in
25   diskette.   These diskettes can be distributed to end users.   The diskettes can
be loaded into hard disk 638.   If other cells want to use one or more of these
databases, the cells can invoke the corresponding drivers (i.e., cells).
Alternatively, these new drivers can also invoke other cells.

The invention has been described with reference to a specific exemplary
30   embodiment thereof.   Various modification and changes may be made
thereunto without departing from the broad spirit and scope of the invention.
The specification and drawings are, accordingly, to be regarded in an
illustrative rather than a restrictive sense; the invention is limited only by the
provided claims.

CLAIMS

1. A database system comprising:

at least a first and a second single-association database (704,705,706);

at least a first and a second database driver (712,713,714) associated with

5   said first and said second databases, respectively;

said first database driver being designed to perform searches on said first database;

said second database driver being designed to perform searches on said second database; and

10 a software module (720) interacting with said first and said second drivers to achieve a desired search result.

2. The database system of claim 1 wherein said first database comprised pairs of characters separated by a predetermined association symbol.

3. The database system of claim 1 wherein said searches on said first

15 and said second databases include logic operation on data in said databases.

4. The database system of claim 1 wherein said software module causing said first and said second database drivers to perform searches on their respective databases and combining results of said searches to achieve said desired search result.

20 5. A network database system including a plurality of computers coupled to a network (812), comprising:

a first single-association database (704,705,706);

a first driver (712,713,714) designed to perform searches on said first database;

25 a second single-association database(704,705,706);

a second database driver (712,713,714) designed to perform searches on said second database;

said first single-association database, said second single-association database, said first database driver and said second database driver residing

30 on different computers on said network;

means for downloading said first single-association database, said second single-association database, said first database driver and said second database driver to a selected computer on said network, if said first single-

association database, said second single-association database, said first database driver and said second database driver are not already residing on said selected computer; and

5   a software module (720) on said selected computer for interacting with said first and said second drivers to achieve a desired search result.

6.   A method for searching data in a database system, comprising:

providing at least a first and a second single-association database;

providing at least a first and a second database driver associated with said first and said second databases, respectively, said first and said second
10   database drivers being designed to perform searches on said first and said second databases, respectively;

providing a software module interacting with said first and said second drivers to achieve a desired search result;

sending a first search instruction to said first database driver by said
15   software module, said first search instruction including a first search criteria;

searching said first single-association database by said first database driver in accordance with said first search criteria to generate a first search result;

sending a second search instruction to said second database driver by
20   said software module, said second search instruction including a second search criteria;

searching said second single-association database by said second database driver in accordance with said second search criteria to generate a second search result; and

25   combining said first and said second search results by said software module to obtain a desired search result.
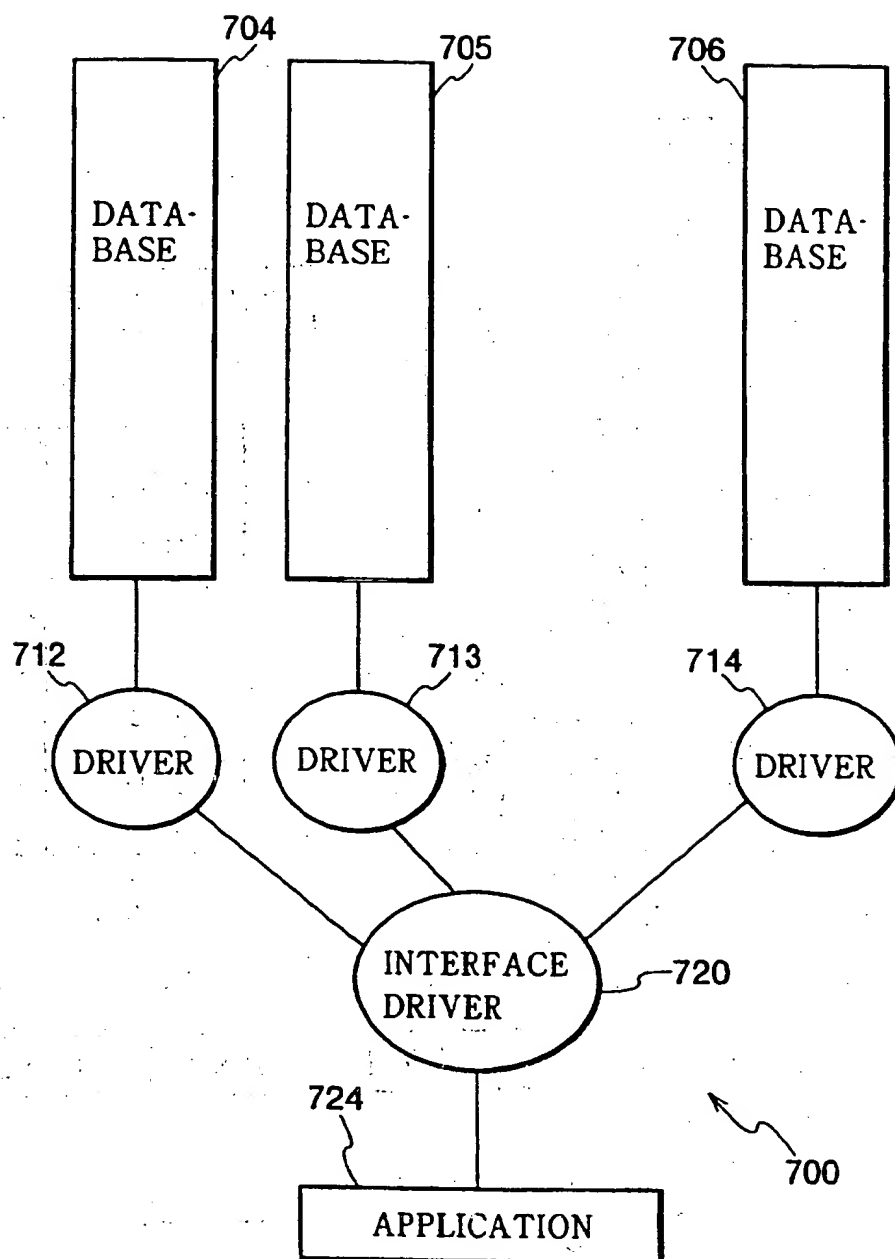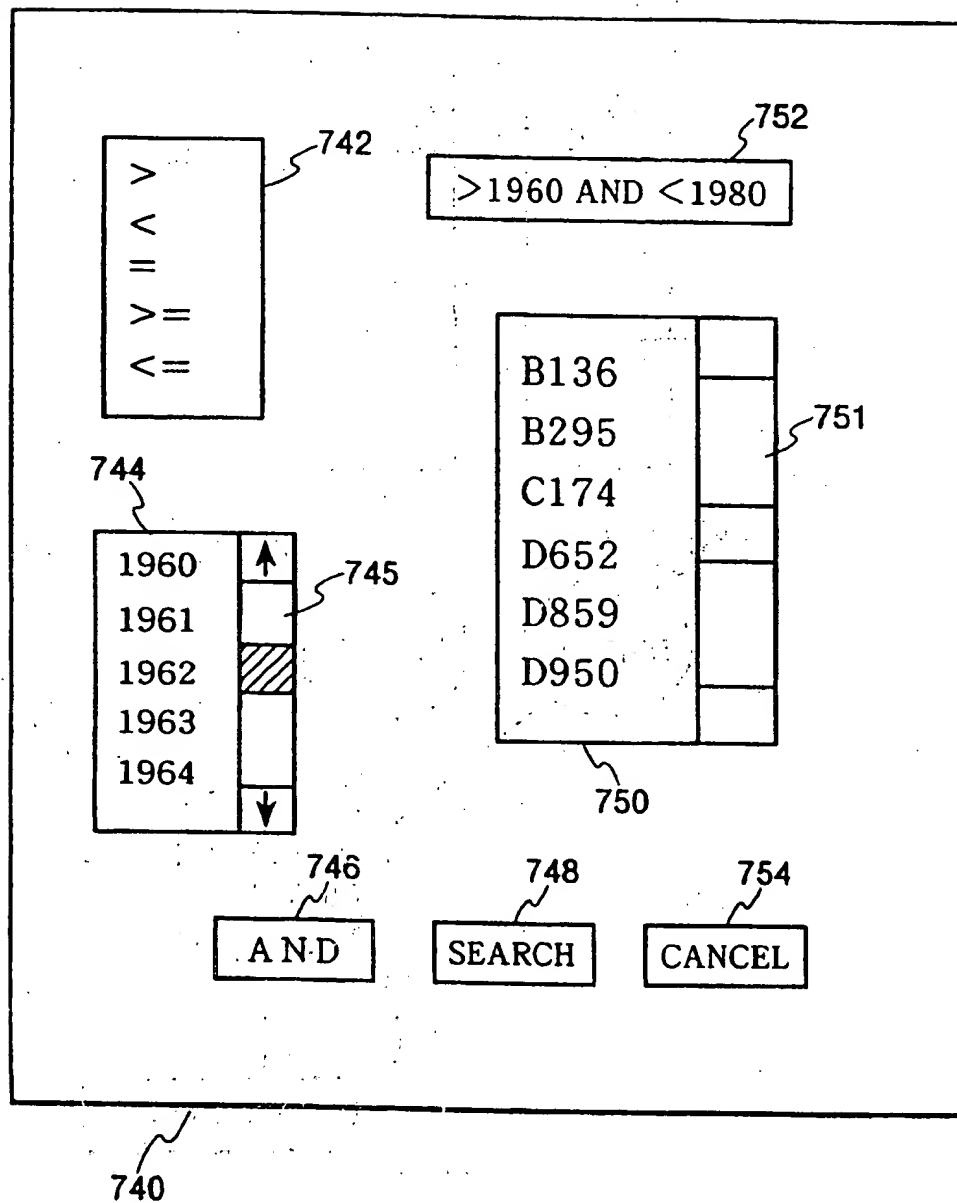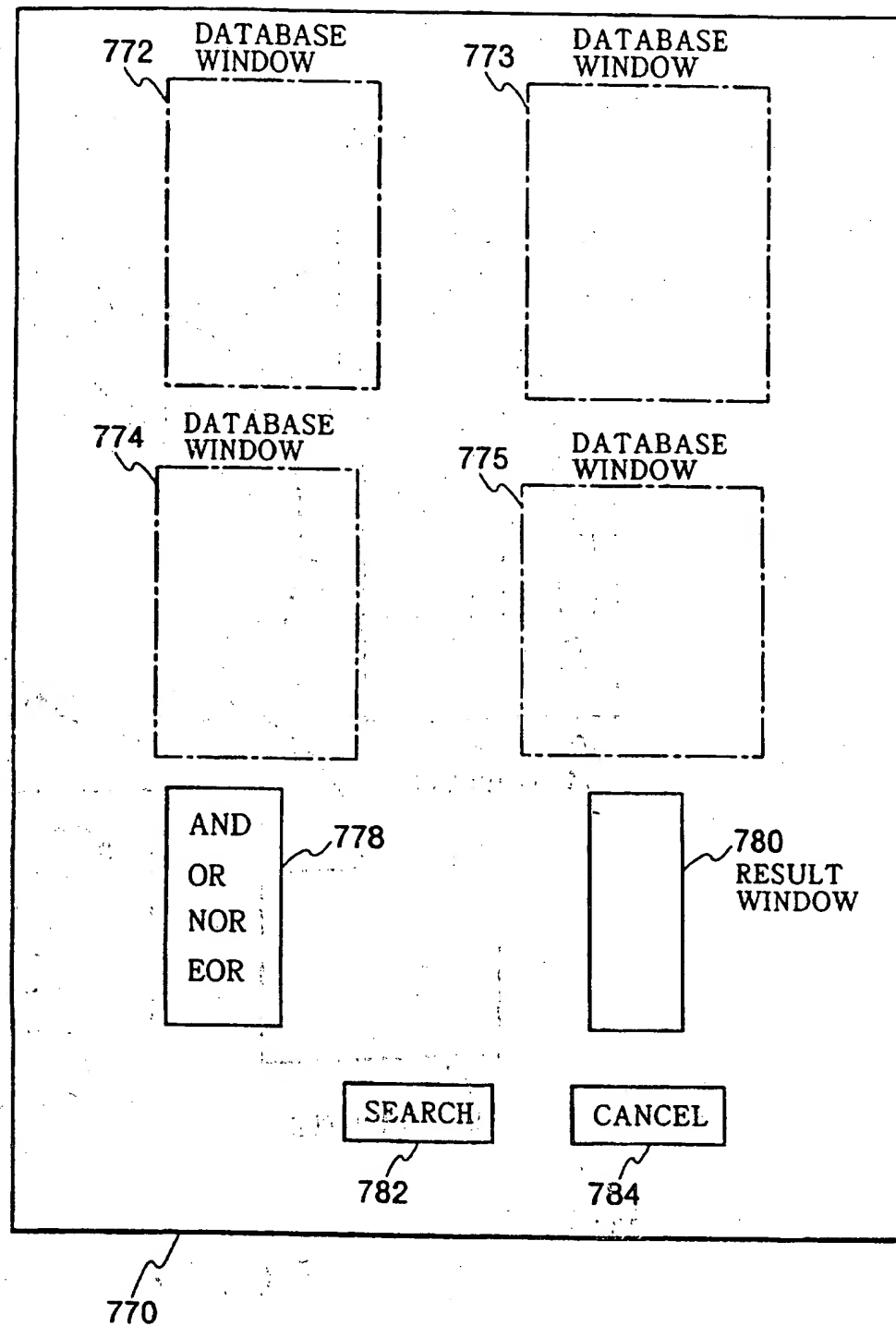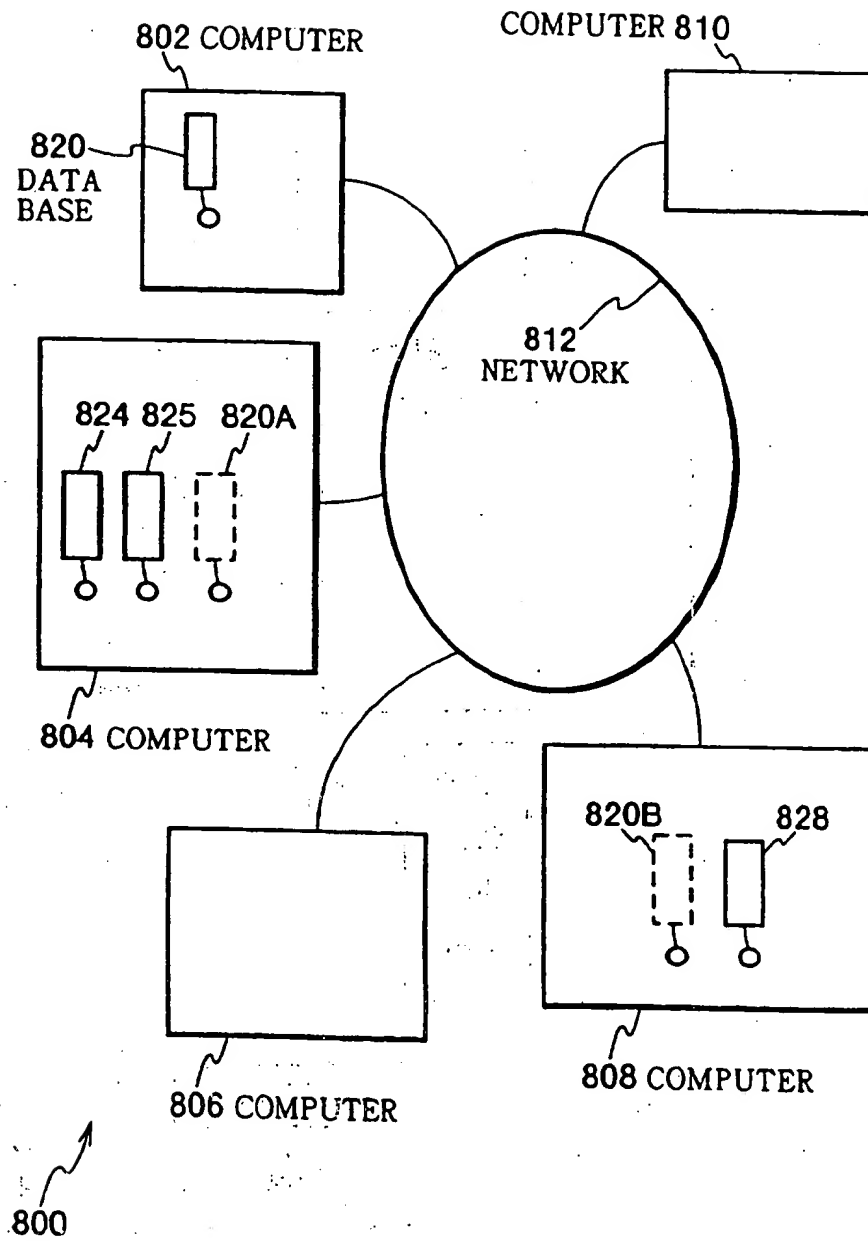
FIG. 1
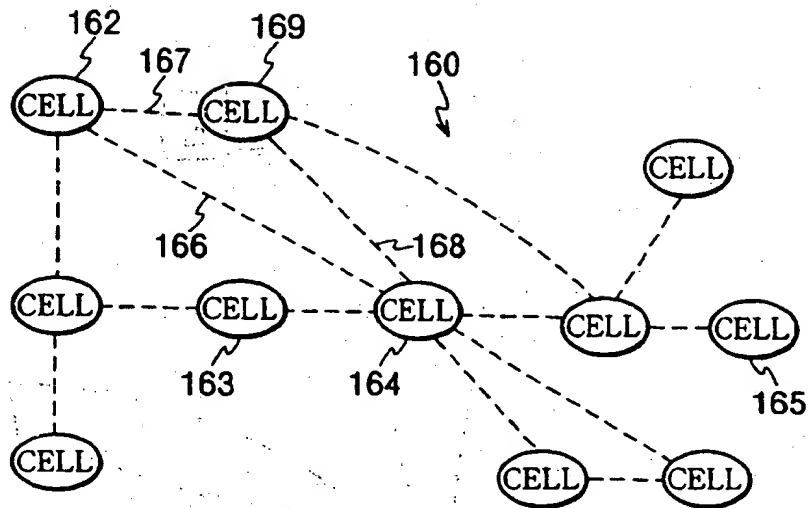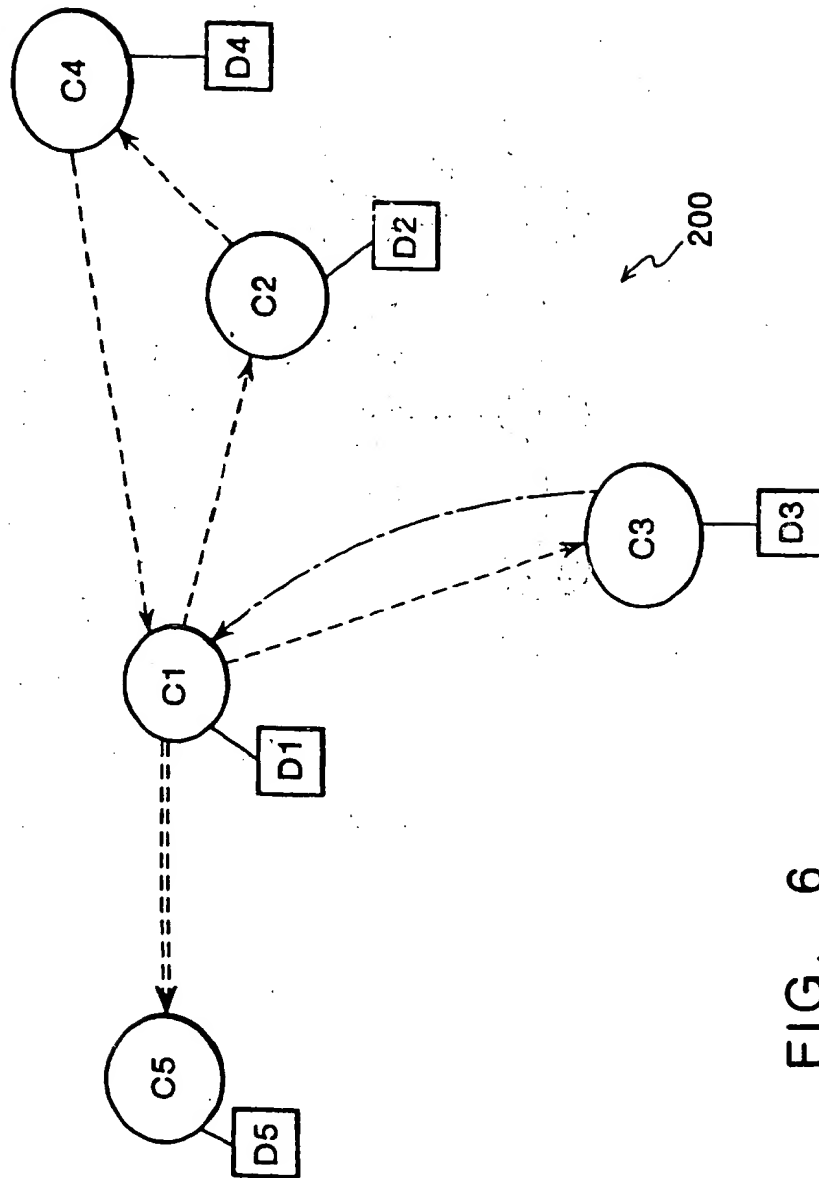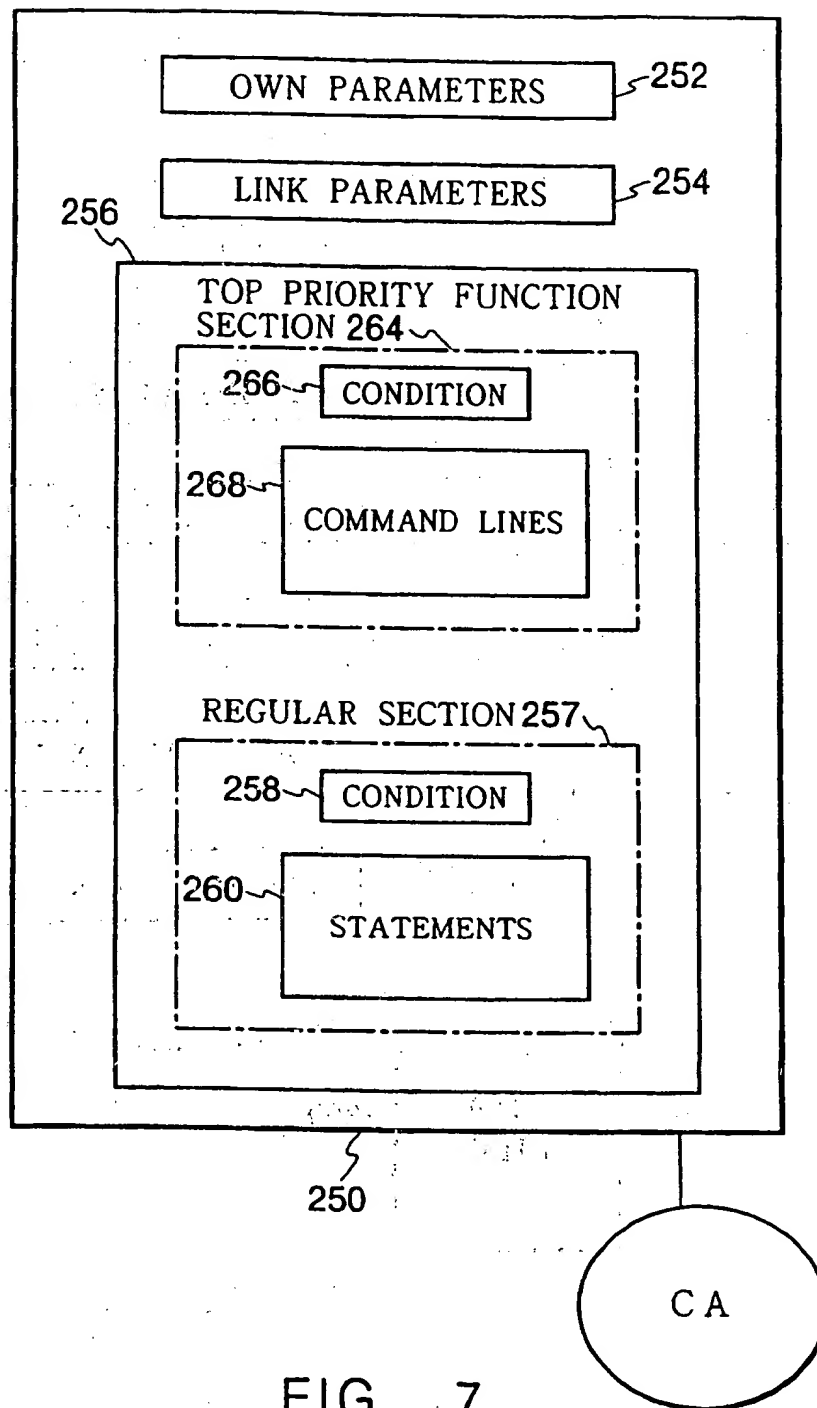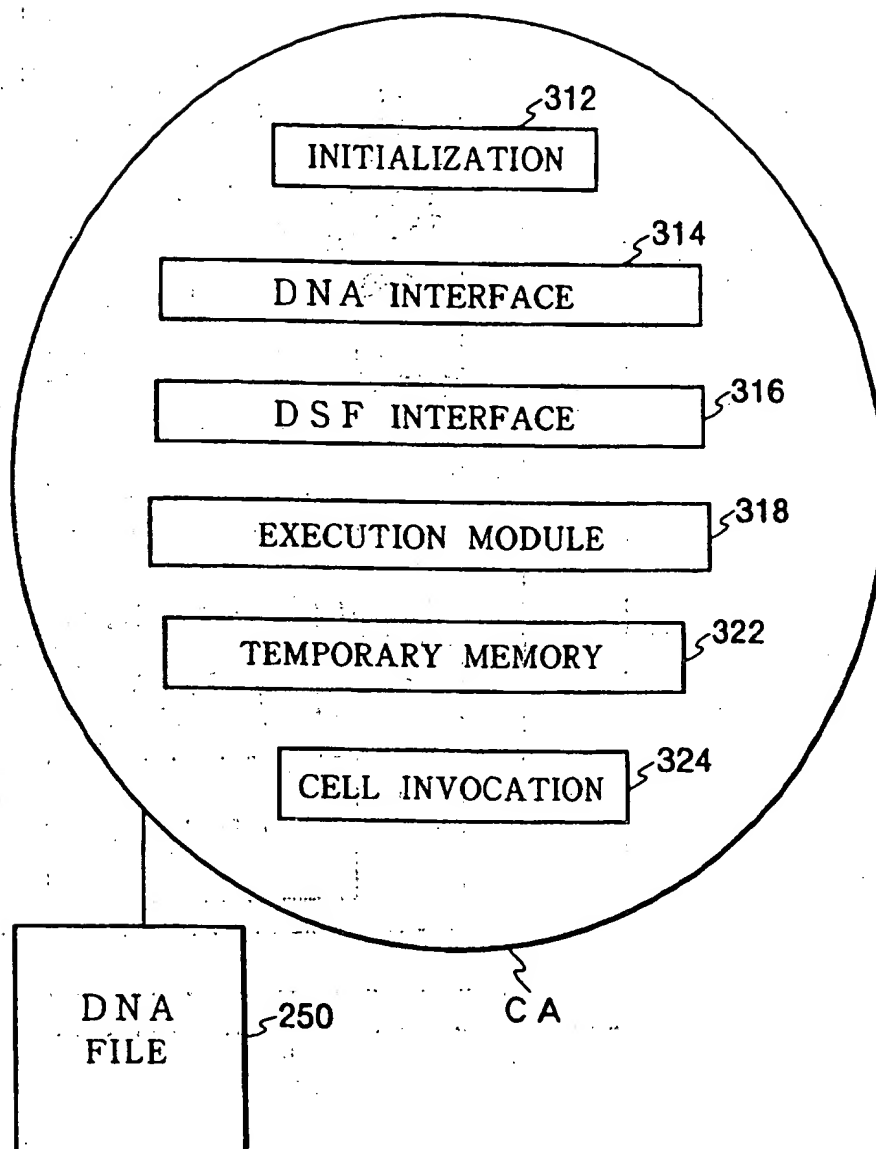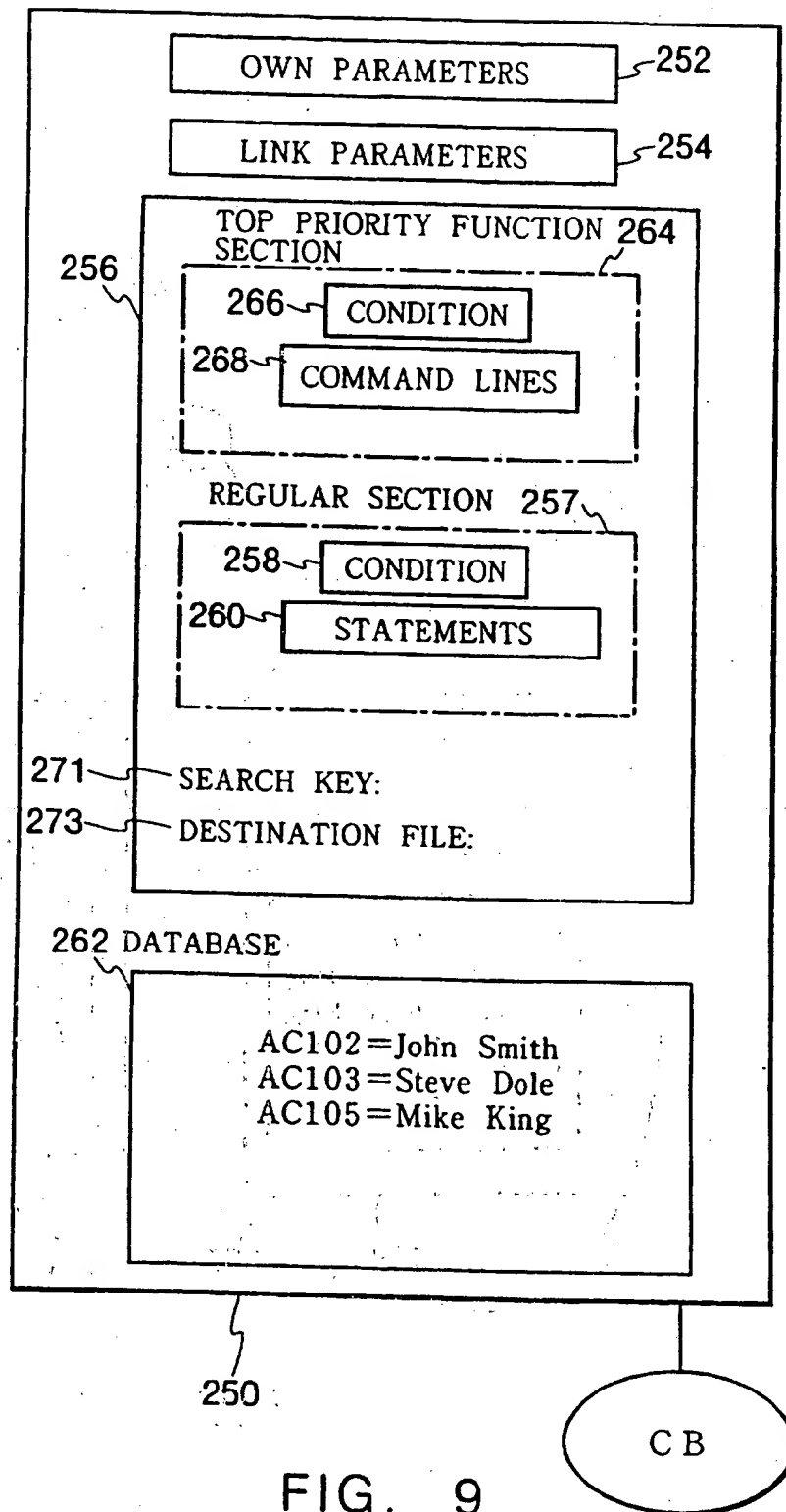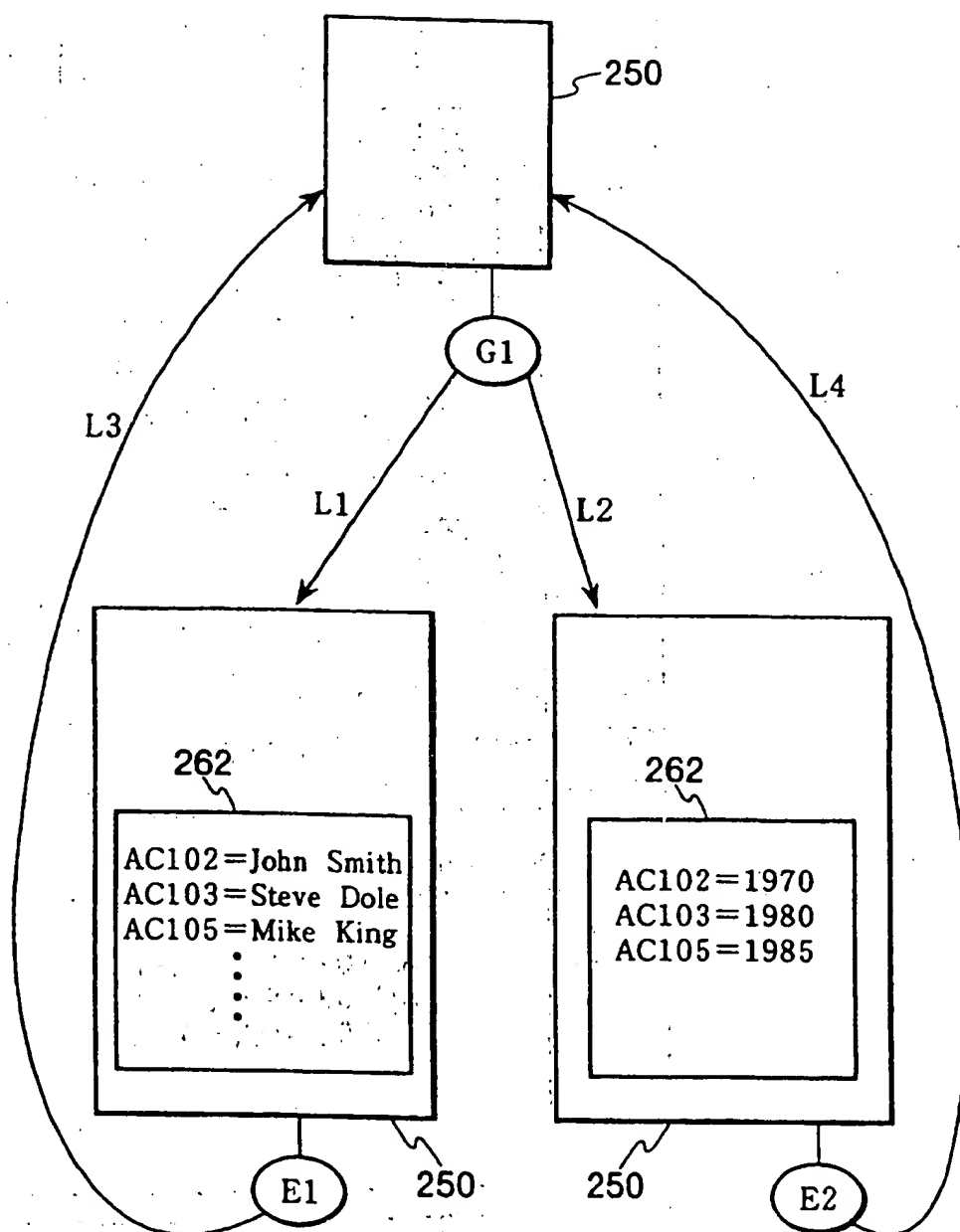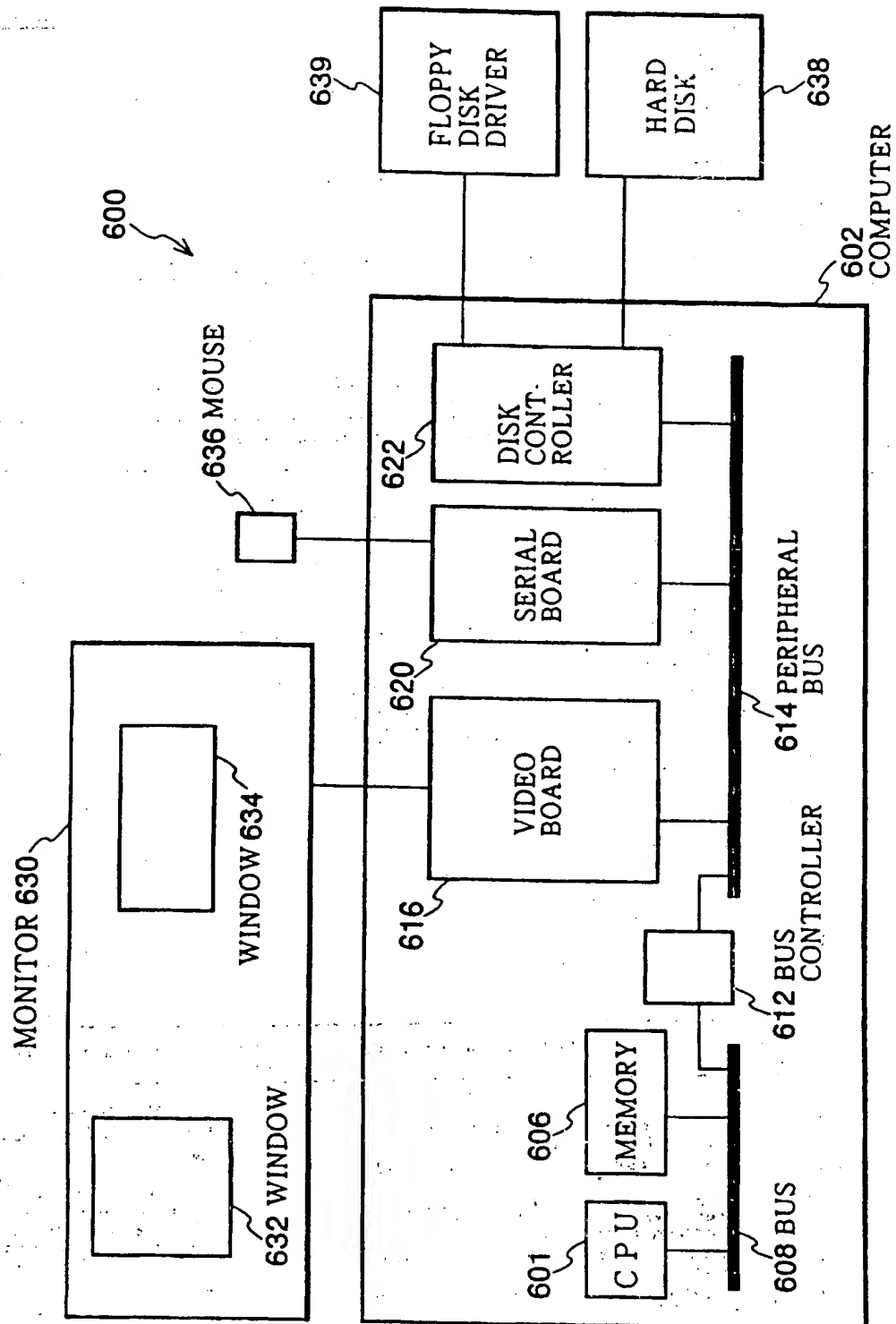
> >
> <
> >=
> <=

742

>1960 AND <1980   752

B136
B295
C174
D652
D859
D950

751

750

744

1960   ↑
1961
1962   ▨
1963
1964   ↓

745

746   748   754

AND   SEARCH   CANCEL

740

FIG. 2

DATABASE
WINDOW

772

DATABASE
WINDOW

773

DATABASE
WINDOW

774

DATABASE
WINDOW

775

AND
OR
NOR
EOR

778

780
RESULT
WINDOW

SEARCH

782

CANCEL

784

770

FIG. 3

FIG. 4

FIG. 5

FIG. 6

● ●

OWN PARAMETERS — 252

LINK PARAMETERS — 254

256

TOP PRIORITY FUNCTION
SECTION 264

266 — CONDITION

268 — COMMAND LINES

REGULAR SECTION 257

258 — CONDITION

260 — STATEMENTS

250

C A

FIG. 7

INITIALIZATION ~312

DNA INTERFACE ~314

DSF INTERFACE ~316

EXECUTION MODULE ~318

TEMPORARY MEMORY ~322

CELL INVOCATION ~324

DNA FILE ~250

CA

FIG. 8

FIG. 9

FIG. 10

FIG. 11

INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER

IPC 6   G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 6   G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category°	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 90 08360 A (TELEBASE SYSTEMS INC) 26 July 1990 see abstract see page 1, line 1 - page 9, line 6 ---	1-6
A	EP 0 690 398 A (CASIO COMPUTER CO LTD) 3 January 1996 see page 2, line 1 - page 3, line 5 ---	1-6
A	EP 0 550 368 A (IBM) 7 July 1993 see abstract -----	1-6

☐ Further documents are listed in the continuation of box C.   ☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

6 June 1997

Date of mailing of the international search report

17. 06.97

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl,
Fax (+ 31-70) 340-3016

Authorized officer

Katerbau, R

Form PCT/ISA/210 (second sheet) (July 1992)

1

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9008360 A	26-07-90	AU 5024990 A	13-08-90
EP 0690398 A	03-01-96	JP 8016619 A	19-01-96
		CN 1121217 A	24-04-96
EP 0550368 A	07-07-93	JP 7191888 A	28-07-95